***FUNCIÓN PARA OBTENER EL HASH***

def get\_hash(key):

    h = 0

    for char in key:

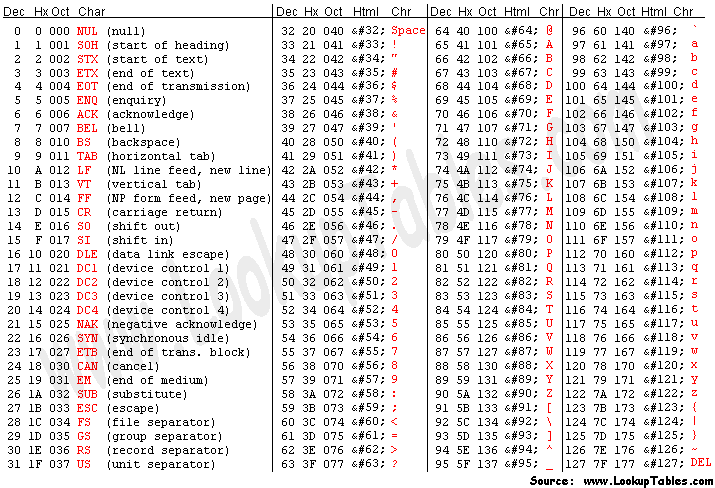
        h += ord(char)

    return h % 100

print(get\_hash("a"))



UNICODE: es un Sistema que asigna un valor numérico a cualquier carácter str (incluyendo letras, dígitos, signos de puntuación, caracteres especiales, entre otros.)



La función ORD(): es una función que toma un STR y te devuelve su valor UNICODE.

***APLICACIÓN DENTRO DE UNA HASH TABLE:***

range = [None for i in range(5)]

print(range)



***CREACION DE LA HASH TABLE***

class HashTable:

    def \_\_init\_\_(self):

        self.max = 100 # CAPACIDAD DE LA HASH TABLE

        self.arr = [None for i in range(self.max)] # CREACION DE LA LISTA CON 100 ELEMENTO DE NONE

***FUNCION PARA OBTENER EL INDICE ATRAVES DEL HASH***

    def get\_hash(self, key):

        h = 0

        for char in key:

            h += ord(char)

        return h % 100

***FUNCION PARA AGREGAR***

    # FUNCION PARA AGREGAR

    def add(self, key, val):

        h = self.get\_hash(key) #KEY = CLAVE PARA BUSCAR EL HASH Y GENERAR EL INDICE

        self.arr[h] = val      #VAL = VALOR QUE QUEREMOS AGREGAR EN EL INDICE KEY

        print(h)

NOTA: ESTO SOLO ES UNA ASIGNACION DE VALORES A UNA LISTA BASICA.

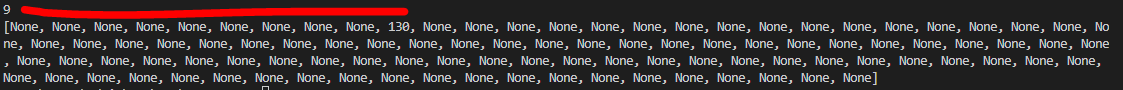
t = HashTable ()

t.add ("march 6", 130)

print(t)

"march 6" = 9 (RESULTADO DE LA FUNCION GET\_HASH)

El valor de 130 se agregó en la posición 9 (0 - 9) de la lista con una capacidad de 100 elementos.



***FUNCION PARA OBTENER UN VAL ATRAVES DE UN INDICE KEY***

    def get(self, key):

        h = self.get\_hash(key)

        return print(self.arr[h])

t.get("march 6")

print(t)

EL VALOR QUE GENERO ES 130.

Texto

Descripción generada automáticamente

***FUNCION PARA IMPRIMIR LA LISTA***

    def \_\_str\_\_(self):

        return str(self.arr)

***ALTERNATIVAS A LA FUNCION GET Y ADD***

    # FUNCION PARA AGREGAR

    def \_\_setitem\_\_(self, key, val):

        h = self.get\_hash(key) #KEY = CLAVE PARA BUSCAR EL HASH Y GENERAR EL INDICE

        self.arr[h] = val      #VAL = VALOR QUE QUEREMOS AGREGAR EN EL INDICE KEY

        print(h)

    def \_\_getitem\_\_(self, key):

        h = self.get\_hash(key)

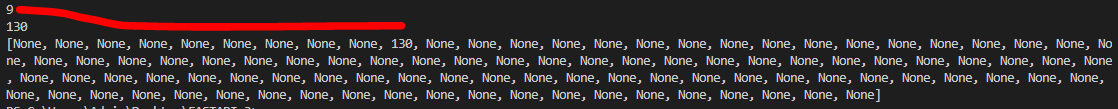
        return print(self.arr[h])

***\_\_SETITEM\_\_***

La diferencia con la función que creamos con GET, es que esta tiene otra forma de enviar los valores y de ser llamada, como, por ejemplo.

t["march 6"] = 130

el resultado sigue siendo el mismo.



***\_\_GETITEM\_\_***

Lo mismo que lo anterior.

t["march 6"]



***ELIMINAR UN ELEMENTO:***

    def \_\_delitem\_\_(self, key):

        h = self.get\_hash(key)

        self.arr[h] = None

del t ["march 6"]

print(t)

A = ANTES

B = DESPUES

Patrón de fondo

Descripción generada automáticamente con confianza baja

***Collision***

Nota, se hizo un cambio en la creación de las 100 listas, ya no es None

    def \_\_init\_\_(self):

        self.max = 100 # CAPACIDAD DE LA HASH TABLE

        self.arr = [[] for i in range(self.max)] # CREACION DE LA LISTA CON 100 ELEMENTO DE NONE

En este caso vamos a sustituir el valor anterior por el valor siguiente o el nuevo.

En la primera iteración, que es con el valor [('march 6', 130)], no se ejecutara el FOR porque la lista está vacía, por ende, se ejecutara el IF NOT FOUND, y se agregara a la lista.

En la segunda iteración se ejecutará el FOR y [('march 6', 130)] pasará a ser [('march 6', 78)].

En la última iteración tampoco se ejecutará el FOR si no que se agregara como la primera iteración, por el IF NOT FOUND.

    def \_\_setitem\_\_ (self, key, val):

        h = self.get\_hash(key)

        found = False

# si ya hay un elemento ([('march 6', 130)]), el idx es 0.

        for idx, element in enumerate(self.arr[h]):

            if len(element) == 2 and element [0] == key:

                a = self.arr[h][idx] = (key, val)

                print(a)

                found = True

                break

        if not found:

            b = self.arr[h]. append ((key, val))

            print(b)

t = HashTable()

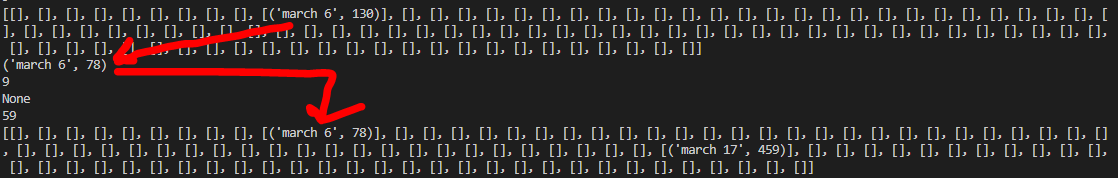
t["march 6"] = 130

print(t)

t["march 6"] = 78

t["march 17"] = 459

print(t)



NOTA: ten en cuenta que el resultado de la lista puede variar dependiendo de donde se ejecute el código de una tabla HASH, ya que las tablas HASH no aseguran el orden si no que pueden tener variaciones potenciales.

Texto

Descripción generada automáticamente

Nota: en este código de agregar un valor, yo he agregado un nuevo valor.

t["hcram 6"] = 25

si te fijas es diferente al:

t["march 6"] = 130

t["march 6"] = 78

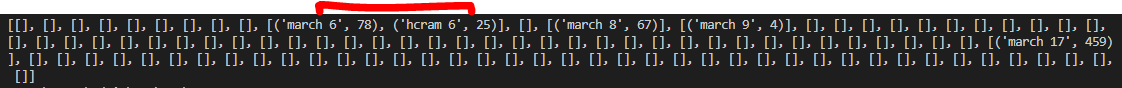
pero ambos producen el índice 9, que significa esto, que el valor:

t["hcram 6"] = 25

se va a agregar al lado de:

t["march 6"] = 78

porque las claves, o los nombres no son iguales.



Al contrario, con:

t["march 6"] = 130

t["march 6"] = 78

que lo que hace es sustituir el 130 por el 78, porque cuando se ejecuta el FOR:

        for idx, element in enumerate(self.arr[h]):

            if len(element) == 2 and element[0] == key:

                self.arr[h][idx] = (key, val)

                found = True

se hace una actualización de valores.

LO SIGUIENTE ES PARA CONSEGUIR UN ELEMENTO DE LOS QUE SE AGREGARON AL MISMO MODULO

    def \_\_getitem\_\_(self, key):

        h = self.get\_hash(key)

        for element in self.arr[h]:

            if element[0] == key:

                return print(element[1])

sí ejecutamos este código, el resultado será el valor porque dentro de la TUPLA que contiene (march 16, 78), el que ocupa el valor 1 (print(element[1])) es el 78.

Si hubiese un 0en (print(element[0])) el resultado sería el texto y no el número.

t["march 6"]

Pantalla de computadora

Descripción generada automáticamente con confianza baja

t["hcram 6"]

Texto

Descripción generada automáticamente

PARA ELIMIAR UN VALOR EN COLISION:

    def \_\_delitem\_\_(self, key):

        arr\_index = self.get\_hash(key)

        for index, kv in enumerate(self.arr[arr\_index]):

            if kv[0] == key:

                del self.arr[arr\_index][index]

en este caso vamos a borrar este:

NOTA: esto toma en cuenta que el valor en la posición 0 es igual al “march 6”.

del t["march 6"]

Texto

Descripción generada automáticamente

Esto funciona para números individuales y en colisión porque están encerrados en una lista [] y esto hace que los Loop FOR iteren sobre la lista.

EJERCICIOS